

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-040414

(43)Date of publication of application : 13.02.1998

(51)Int.Cl.

G06T 15/00
G06F 15/16
G06T 1/20
// G06T 1/60

(21)Application number : 08-210577

(71)Applicant : HITACHI LTD

(22)Date of filing : 22.07.1996

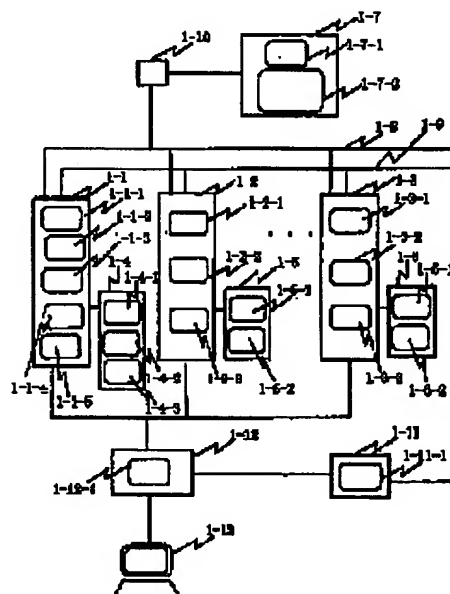
(72)Inventor : SUGITA YUMIKO
KIMURA SHINJI
KUWANA TOSHIYUKI

(54) MULTIPROCESSOR PLOTTING AND PROCESSING DEVICE

(57)Abstract:

PROBLEM TO BE SOLVED: To reduce the time for access in the case of referring to data without being limited by the capacity, data amount or data constitution of respective local memories in the case of ray-casting system calculation.

SOLUTION: A shared memory 1-7 stores three-dimensional numerical data and information concerning the constitution of data and a data reference pattern and a processor 1-1 divides processing and distributes processing to processors from 1-1 to 1-3. The respective processors are parallelly operated, perform processing for finding the arrangement information and buffer sizes of reference data at the time of calculation respectively, secure buffers in local memories from 1-4 to 1-6, store the arrangement information or the buffer sizes in management information tables inside the local memories and afterwards generate two-dimensional images by executing processing partially charged in plotting processing. When the buffers can not be secured because local memories are lacked, it is reported to the processor 101, this processor refers to the working condition information of respective processors, divides processing again and transmits it to the respective processors and based on this transmission, the respective processors execute processing.



LEGAL STATUS

[Date of request for examination] 28.02.2000

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]	3292976
[Date of registration]	05.04.2002
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平10-40414

(43)公開日 平成10年(1998) 2月13日

(51)Int.Cl. ⁹	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 T 15/00			G 0 6 F 15/72	4 5 0 K
G 0 6 F 15/16	3 8 0		15/16	3 8 0 Z
G 0 6 T 1/20			15/66	K
// G 0 6 T 1/60			15/64	4 5 0 F

審査請求 未請求 請求項の数4 F D (全 15 頁)

(21)出願番号 特願平8-210577

(22)出願日 平成8年(1996) 7月22日

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72)発明者 杉田 由美子

神奈川県川崎市麻生区王禅寺1099番地 株

式会社日立製作所システム開発研究所内

(72)発明者 木村 信二

神奈川県川崎市麻生区王禅寺1099番地 株

式会社日立製作所システム開発研究所内

(72)発明者 桑名 利幸

茨城県日立市大みか町五丁目2番1号 株

式会社日立製作所大みか工場内

(74)代理人 弁理士 笹岡 茂 (外1名)

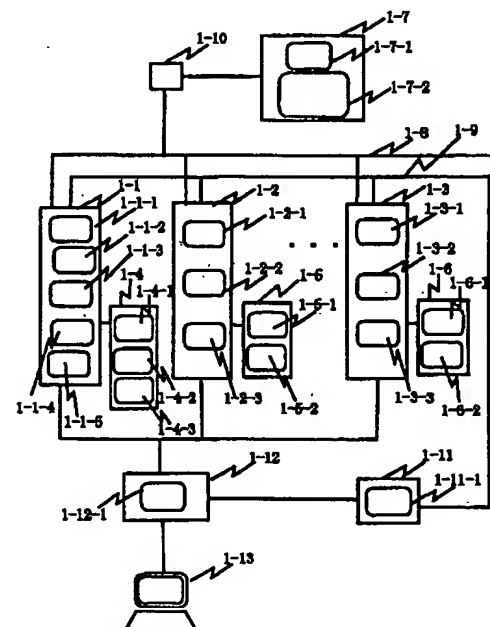
(54)【発明の名称】 マルチプロセッサ描画処理装置

(57)【要約】

【課題】 レイキャスティング方式の計算において、各ローカルメモリの容量やデータ量、データ構成に制限されずにデータ参照時のアクセス時間を減らす。

【解決手段】 共有メモリ1-7には3次元数値データとデータの構成とデータ参照パターンに関する情報が格納され、プロセッサ1-1は処理を分割し、プロセッサ1-1～3に処理の分配を行う。各プロセッサは並列動作し、夫々計算時の参照データの配置情報とバッファサイズを求める処理をし、ローカルメモリ1-4～6内にバッファを確保し、配置情報やバッファサイズはローカルメモリ内の管理情報テーブルに格納し、その後、描画処理で分担した処理を実行して2次元画像を生成する。ローカルメモリ不足でバッファ確保できない場合には、それをプロセッサ1-1に通知し、該プロセッサはプロセッサの稼働状況情報を参照し、処理を再分割して各プロセッサへ送信し、これに基づき各プロセッサは処理を実行する。

図 1



【特許請求の範囲】

【請求項1】 大容量の3次元数値データを格納する手段と、複数のプロセッサと、前記数値データ格納手段とは独立に前記プロセッサ個々に割り当てられたローカルメモリとで構成し、大容量3次元数値データを用い、前記複数プロセッサがそれぞれ担当処理範囲の情報に従いレイキャスティング法に基づく計算により2次元画像を生成して可視化する描画処理装置において、一連の前記計算で参照するデータの配置情報を収集する手段と、

前記数値データ格納手段から読み込んだデータを重ならず格納するバッファサイズを前記収集する手段を用いて算出し前記ローカルメモリに該バッファサイズのバッファを確保する手段と、

前記計算をする際に参照する前記3次元数値データを前記ローカルメモリにバッファに格納し、2回目以降の参照時にはデータがすでにローカルメモリのバッファに存在するかを識別し、存在すればローカルメモリのバッファから読み込む手段と、を前記各プロセッサが有し、各プロセッサが並列に動作することを特徴とするマルチプロセッサ描画処理装置。

【請求項2】 請求項1記載のマルチプロセッサ描画処理装置において、

前記バッファを確保する手段は、前記ローカルメモリの空き領域サイズを得てその範囲内でバッファサイズを求めることを特徴とするマルチプロセッサ描画処理装置。

【請求項3】 請求項2記載のマルチプロセッサ描画処理装置において、

前記複数のプロセッサのいずれか1台のプロセッサを全プロセッサの稼働状態を管理するプロセッサとし、各プロセッサは、ローカルメモリにバッファを確保できない場合に前記管理プロセッサに稼働不可能の通知と前記担当処理範囲の情報を伝える通信手段を有し、通知を受けた前記管理プロセッサは、前記稼働不可能なプロセッサ担当分の処理を稼働可能な他のプロセッサに分配する手段を有することを特徴とするマルチプロセッサ描画処理装置。

【請求項4】 請求項1記載のマルチプロセッサ描画処理装置において、

前記各プロセッサは、1回目の実行時にレイキャスティング法での各ピクセルに対する計算時に参照した前記3次元数値データの奥行き方向の最大処理幅を得る手段と、

同じデータでの2回目の実行時に該手段で得た値を用いてバッファサイズを再計算を行ない、得たサイズが前回よりも小さい場合にはバッファを再確保する手段を有することを特徴とするマルチプロセッサ描画処理装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は画像及びCG描画処

理システムに係わり、特にポリウムレンダリング処理やレイトレーシング処理など、レイキャスティング方式で3次元数値データから2次元画像を算出する処理に好適なマルチプロセッサ描画処理装置に関する。

【0002】

【従来の技術】 画像の分野では最近、医療分野におけるMRIやCTデータを利用したポリウムレンダリング処理や3次元CGにおける高品質機能を実現するレイトレーシング処理が適用されている。ポリウムレンダリングとはモデルの表面を対象とするのではなく、測定結果から得られたデータが表現されている3次元離散データのボクセルが持っている数値データ（温度、密度、強度など）と輝度に応じた色を、視線トレースに沿って2次元のピクセル空間に累積写像して表示するものである。なお、ボクセルとは3次元空間をx、y、z方向に小さな単位の格子に分割して構成したもので、対象となるデータがこの格子内にあれば1、無ければ0の値で表現される。ポリウムレンダリングの詳細な手法に関しては「ポリウム・ビジュアライゼーションの基本アルゴリズム」PIXEL NO. 121 PP130-137に記述されている。

【0003】 このポリウムレンダリングにおいて高精度／高品質／高画質な画像を得るためには、1ボクセルを求めるのに複数のデータを参照して正規化や輝度の計算を行なわねばならないため、全ピクセルを計算するには参照するデータは膨大で、処理全体で参照するデータ数が数メガバイトのオーダーになることも珍しくない。よりリアルで正確な高品質画像を得ようとすればするほど、参照するデータ量や参照回数は多くなり、その分実行時間がかかる。バスも高速になりメモリも大容量になってはきているが、価格や実装上の構成の問題もあり、まだ低価格では実用的な性能を出すに至っていない。

【0004】 この実行時間を短縮する方法として、以下のような参照するデータの参照回数や参照方法を工夫する技術が考えられている。マルチプロセッサ構成のシステムにおいては、対象のデータをマルチプロセッサ数で分割し各プロセッサのローカルメモリに各々転送、各プロセッサはローカルメモリ内のデータのみで処理をし、他のプロセッサのメモリ内のデータでの処理が生じたらそのデータを持つプロセッサに処理を渡し、少ないメモリ量での処理を行ない並列処理による高速化を行なう方式が特開平6-274647「ローカルメモリ型並列可視化装置」に示されている。対象のデータ群すべては共有メモリに格納しておき、処理で参照するデータのアクセスを一度トレースし、トレース情報を保管し、2回目からの実行ではこのトレース情報を基に共有メモリからデータをキャッシュに先読みして、メモリアccessのオーバーヘッドを削減し高速化を行なう方式が特開平6-324942「並列計算機システム」に示されている。しかしレイトレーシングやポリウムレンダリングのよう

3

に、膨大な量のデータを参照し、3次元空間内の任意の視点からの可視化が要求されるシステムにおいては、これらの方式ではまだ課題が多い。

【0005】マルチプロセッサでデータを均等に分割分配し負荷を分散させ、各プロセッサでそのデータに対してのみ処理を行なう方式では、各プロセッサに分割後のデータすべてを格納できるサイズ分のメモリがあることが条件であり、任意の1つのプロセッサが他の処理でローカルメモリを用いていた場合など、分割データがローカルメモリに格納しきれない時の事は考えられていない。また、3次元数値データには描画処理が参照しないデータも多くあり、その分のデータ転送は処理のオーバーヘッドとなる。さらに他のプロセッサのデータを使用する場合にはプロセスを移す必要があるが、視点角度によっては何度も行なわなくてはならず、通信のオーバーヘッドが多くなりかえって時間がかかる可能性がある。また、トレースを用いた方式では、データの内容や表示条件によっては視点角度が違おうと参照するデータが異なるため、事前に得たトレース情報が役に立たない場合が多い。さらに常に高速に実行するためにはあらゆる条件に対するトレース情報を収集し管理しておく必要があり、この情報量は無視できない大きさになる。

【0006】

【発明が解決しようとする課題】膨大なデータの参照処理が多い描画処理を行なうマルチプロセッサシステムにおける課題には、以下の項目がある。第1の課題は、参照データの容量やプロセッサ数、各プロセッサのローカルメモリの容量に制限されることなく、さらには無駄なデータ転送を行わずに、データのアクセス時間を減らす仕組みをローカルメモリのバッファを用いて実現し、高速に処理を実行することである。第2の課題は、任意のプロセッサがローカルメモリ不足で実行不可能である場合でも、処理を続行できるようにすることである。第3の課題は、一度行なった描画処理から参照データのトレースなど大量の情報を収集するのではなく、少ない情報量で処理に適したバッファサイズを得て最適化と高速化を図ることである。

【0007】

【課題を解決するための手段】上記課題を実現するため、本発明は、大容量の3次元数値データを格納する手段と、複数のプロセッサと、前記数値データ格納手段とは独立に前記プロセッサ個々に割り当てられたローカルメモリとで構成し、大容量3次元数値データを用い、前記複数のプロセッサがそれぞれ担当処理範囲の情報に従いレイキャスティング法に基づく計算により2次元画像を生成して可視化する描画処理装置において、一連の前記計算で参照するデータの配置情報を収集する手段と、前記数値データ格納手段から読み込んだデータを重ならず格納するバッファサイズを前記収集する手段を用いて算出し前記ローカルメモリに該バッファサイズのバッ

4

ァを確保する手段と、前記計算をする際に参照する前記3次元数値データを前記ローカルメモリのバッファに格納し、2回目以降の参照時にはデータがすでにローカルメモリのバッファに存在するかを識別し、存在すればローカルメモリから読み込む手段と、を前記各プロセッサが有し、各プロセッサが並列に動作するようにしている。

【0008】さらに、前記バッファを確保する手段は、前記ローカルメモリの空き領域サイズを得てその範囲内でバッファサイズを求めるようにしている。

【0009】また、前記複数のプロセッサのいずれか1台のプロセッサを全プロセッサの稼働状態を管理するプロセッサとし、各プロセッサは、ローカルメモリにバッファを確保できない場合に前記管理プロセッサに稼働不可能の通知と前記担当処理範囲の情報を伝える通信手段を有し、通知を受けた前記管理プロセッサは、前記稼働不可能なプロセッサ担当分の処理を稼働可能な他のプロセッサに分配する手段を有するようにしている。

【0010】また、前記各プロセッサは、1回目の実行時にレイキャスティング法での各ピクセルに対する計算時に参照した前記3次元数値データの奥行き方向の最大処理幅を得る手段と、同じデータでの2回目の実行時に該手段で得た値を用いてバッファサイズを再計算を行ない、得たサイズが前回よりも小さい場合にはバッファを再確保する手段を有するようにしている。

【0011】

【発明の実施の形態】以下、本発明の実施の形態の一例を詳細に説明する。図1は、本発明を適用するシステムの構成図である。ルートプロセッサ1-1はユーザからの描画要求を受け付けると、3次元数値データ1-7-1とデータの構成とデータ参照パターンに関する情報1-7-2を共有メモリ1-7へ格納し、描画条件の取得し、処理の分配を行なう機能1-1-5、を実行する。分割した処理に関する情報は、ルートプロセッサからバス1-9を経由して複数のプロセッサ群1-2～3にデータ転送する。複数のプロセッサ群1-2～3は分割された処理を各々分担し並列に動作する。各プロセッサ1-1～3は共有メモリ1-7からバス1-8を経由してデータの入出力を行なう。この時、複数のプロセッサからの読み出し／書き込み要求はバスブリッジ1-10で制御する。各プロセッサ内では描画処理に先立ち、計算時の参照データの配置情報とバッファサイズを求める処理1-1-1、1-2-1、1-3-1を実行し、ローカルメモリ1-4～6内にバッファ1-4-1、1-5-1、1-6-1を確保する。配置情報やバッファサイズはローカルメモリ1-4～6内にある管理情報テーブル1-4-2、1-5-2、1-6-2に格納する。その後、描画処理1-1-2、1-2-2、1-3-2で分担した処理を実行して2次元画像を生成する。この描画処理の中でデータの視点トレース回数の算出を行ない、ローカルメモリ1-4～6

内にある管理情報テーブル1-4-2、1-5-2、1-6-2に設定する。なおローカルメモリ1-4~6は共有メモリ1-7とは独立に各プロセッサと接続されている。

【0012】バッファ確保処理時、ローカルメモリが不足して確保できない場合には、他プロセッサとの通信処理1-1-3、1-2-3、1-3-3を用いて、ルートプロセッサ1-1に通知する。ルートプロセッサは任意プロセッサからの通信を受け取ると、接続されているプロセッサの稼働状況情報1-4-3を参照し、処理を再分割して任意複数プロセッサへ分配送信する処理1-1-4を実行する。各プロセッサ1-1~3で生成した2次元画像の値1-11-1はフレームバッファ1-11へ格納し、表示処理プロセッサ1-12へ渡った表示命令を表示処理機能1-12-1で処理した時に、画面1-13へ表示する。なお、本実施例では2次元画像を生成する描画計算はボリュームレンダリングを適用する。

【0013】図2は、本発明の要素として各プロセッサが得る計算時の参照データの配置情報とバッファサイズ情報1-4-2、1-5-2、1-6-2の詳細を示したものである。各プロセッサが得る情報には、少なくとも以下の情報を持つ。まず共有メモリ上にある3次元数値データに関して、その先頭アドレス2-1と、3次元データ個々のデータ長2-2と、3次元データ全体のx方向サイズ2-3、3次元データ全体のy方向サイズ2-4、3次元データ全体のz方向サイズ2-5、3次元データのデータ構成識別子2-6を持つ。データ構成識別子については図3で説明する。情報2-2~6は3次元データの構成情報であり、ルートプロセッサ1-1が任意の媒体から3次元数値データを共有メモリに読み込んだ時に識別し、他プロセッサにも通知する。その他には得た情報を共有メモリに格納し各プロセッサが読みに行き得る方法もある。これらの情報はボリュームレンダリング処理でデータの参照位置を制御するのに用いる。本実施例ではバッファサイズの計算材料としても用いる。

【0014】各プロセッサに割り当てられた描画対象領域の情報として、各プロセッサの処理担当領域の個数2-7と、その領域の開始位置のx座標2-8、2-10、開始位置のy座標2-9、2-11を持つ。これらは並列処理を行なうための情報であり、各プロセッサの処理担当部分は1以上の領域から成り、それらの位置を示すすべてのx座標とy座標の情報を持つ。分割はルートプロセッサが行ない、描画命令と一緒に通知されてきたこれらの情報を各プロセッサの管理情報テーブルに格納する。バッファサイズ計算用のバッファのx方向のサイズ2-12、y方向サイズ2-13、z方向サイズ2-14は、計算時に参照するデータの配置情報から得る。これは、全プロセッサで行なう。その他にも、ルートプロセッサが代表して行ない各プロセッサに通知する方法がある。参照データの配置情報とバッファの方向サ

イズと全体サイズの求め方についてはバッファの管理方法とともに図4~8で説明する。

【0015】視線トレースの最大回数、すなわちz方向の参照幅の最大値2-15は描画実行時に算出して格納する。さらにローカルメモリに確保したバッファの先頭アドレス2-16と、そのバッファサイズ2-17と、バッファのx方向サイズ2-18、バッファのy方向サイズ2-19、バッファのz方向サイズ2-20の情報も持つ。バッファの先頭アドレス2-16はバッファの1/O制御時に参照する。これらの情報(2-17~20)は2回目以降に最適なバッファサイズを再計算する際に使用する。

【0016】図3は3次元数値データの構成例である。図3に示した3次元数値データ3-1は1データ単位の構成を採る。各単位データはまずz方向3-2-1に並び、その列がx方向3-2-2に並んでいる。さらに(xサイズ, yサイズ, zサイズ)=(m, 1, n)(ただし、m, n=各方向の最大サイズ、1は数値である)で構成される面はy方向3-2-3に並んでいる。すなわち3次元数値データ3-1のデータ3-1-1~5を直線的な位置表現3-3で表すと、それぞれ矢印で示した対応となり3-3-1~5の位置となる。図3のデータ構成の場合のデータ構成識別子2-6を1と定義する。データの構成には他に、同じ1データ単位でx, y, z方向のデータの並びの優先度が違う構成や、(xサイズ, yサイズ, zサイズ)=(a, a, 1)のsquare型を単位構成として並ぶ構成などがあり、それらにもそれぞれ2以上の一意な正数値を定義する。

【0017】図4は図3の構成における1ボクセルのボリュームレンダリング計算処理での3次元角度(0°, 0°, 0°)(正面から見た場合である)での参照パターン例である。処理対象データの計算では、任意の角度に対しても精密なデータを得るために、対象データの近傍のデータを用いて正規化する。本実施例では例えば対象データを4-1とした時、そのデータを含む周囲8点のデータ4-1~8を参照して計算する方法を採用する。さらにボリュームレンダリングでは反射処理のために対象データの法線ベクトルを求める必要があり、これには4-2~5、4-9~11の位置にあるデータを用いて計算する。それぞれの位置データに対しても任意角度に対応した精度を出すため、そのデータを含む周囲8点を参照して求める。従って、データ4-1を対象とした計算処理において参照するデータとその位置を識別すると、図4に示した32点となり、そのすべての点を含むデータ構成はx, y, z方向サイズがそれぞれ4の立方体構成となる。この値は1回目の計算で参照したデータの座標から各方向の最大値と最少値を求めることにより算出できる。1ボクセルを求める計算ではこの32点のうち8点は2回以上参照しており、一度読み込んだこれらのデータをより高速にアクセスできる場所に保管し

ておければ2回目以降の参照は高速に行なえることから、容量は小さいがアクセス性能が速いローカルメモリにこのデータを保管するバッファを確保する。このバッファの最少サイズは前述で求めた構成の64でよいが、ローカルメモリに余裕がある場合には、さらに効率の良いサイズを確保する。

【0018】図5は求める画素（ピクセル）と参照するデータ群との一番単純な関係を示したものである。レイキャスティング方式では、画素（ピクセル）5-1~4はそれぞれデータ群5-5~8を図4に示した参照パターンで参照して計算する。すなわちz方向に並ぶデータを対象に計算処理が進む。例えば、図4でデータ4-1の次に計算対象となるのはデータ4-5である。データ4-1と4-5の計算で参照するデータは図4の4*4*4の立方体のデータ構成の中に28個ある。次に計算対象になるのがデータ4-12の場合は20個ある。z方向に1つサイズを大きくした直方体分のデータを読み込めるバッファがあれば、24個になる。従ってz方向に並ぶデータをより多くバッファに格納できれば、再利用できるデータが多くなり効率がよい。図3に示した3次元数値データはz方向に並んでいること、共有メモリからデータを読み込む処理では1回に複数バイト読み込まれることを利用し、共有メモリから1回に読み込むバイト数の倍数をz方向のデータサイズに採用する。例えば1回に読み込むバイト数が16の場合はz方向サイズ2-20は16の倍数とする。

【0019】図6は、確保したバッファに共有メモリのどのデータが入っているかを管理するためのバッファ管理情報テーブルである。バッファの位置を識別するインデックス値6-1と現在そのバッファ位置に入っている3次元数値データの共有メモリ内でのアドレスを識別できる値6-2から成る。例えば、16バイトづつデータがバッファに書き込まれるとすると、index1には最初の16バイトのデータが書き込まれるバッファ内のアドレスが書き込まれ、6-2には、その16バイトデータの共有メモリ内のアドレスが書き込まれ、index2には次の16バイトのデータが書き込まれるバッファ内のアドレスが書き込まれ、6-2には、その16バイトデータの共有メモリ内のアドレスが書き込まれ、以下同様に行われる。このバッファ管理情報テーブルについては図示されていないが、各プロセッサのローカルメモリに用意される。共有メモリアドレス識別子6-2の初期値はすべて-1である。バッファに16バイト単位で格納する場合にはインデックスは（バッファ相対アドレス÷16）で算出する。

【0020】図7は確保したバッファ7-2と共有メモリ内の3次元数値データ7-1との対応を示したものである。共有メモリ内の3次元数値データ7-1の任意のnバイト（nはバッファのz方向サイズ。例えば16）のデータ群7-1-1はバッファ7-2内のエリア7-

2-1に格納する。同様に共有メモリ内の3次元数値データ群7-1-2~7はバッファ7-2内のエリア7-2-2~5に格納する。この対応には対象となる3次元数値データの3次元数値データ先頭からの相対アドレスをバッファサイズで割り、その余りをバッファの相対アドレスとして用いている。従ってこの方法では、共有メモリ内の3次元数値データ群7-1-1と7-1-5がバッファのエリア7-2-1に格納され、3次元数値データ群7-1-3と7-1-7がバッファのエリア7-2-4に格納されるように、同じ位置に格納されることがある。

【0021】そこで、この制御方式でも参照データが重なりあうことなく格納できるバッファサイズを求める処理を提供する。この計算には前述までの説明で求めた参照データの構成に基づくバッファサイズを用いる。図4に示したように隣あうx座標値やy座標値のデータ（例えば4-1と4-2）は参照データの開始/終了のz位置に近いことが多い。従って、バッファに読み込む時には、同じ計算で参照するデータがバッファ内で書きしあわないように、まずバッファサイズはz方向サイズの公約数やz方向とx方向のサイズを掛け合わせた値の公約数を避けた値にする。例えば、もし公約数であれば、バッファのx、y方向サイズを現在の値よりも大きくかつ一番小さな素数値に変更する。3次元数値データの各方向サイズを256とした場合、前述で求めた参照配置情報のx、zの方向サイズ4は公約数であるので、バッファのx、y方向のサイズを現在の値よりも大きくかつ一番小さな素数5に変更する。これを図2のバッファのx、y、z方向サイズ2-18~20の初期値として登録する。これでx方向に並ぶデータが同じバッファ位置に格納されることはない。さらにy方向に並ぶデータにも対処できるように、バッファのx、y、z方向サイズを使って値を求めて加算し、バッファサイズを算出する。この値はバッファサイズ2-17に算出したサイズを格納する。

【0022】図8は算出したバッファサイズに基づくデータの格納結果である。本例では前述で求めたバッファのx、y、z方向サイズを掛け合わせた値に、x、y方向サイズを掛け合わせた値から1を引いた値にz方向サイズを掛けた値を加算して求める。式を以下に示す。

$$(x_Size * y_Size * z_Size) + ((x_Size * y_Size - 1) * z_Size)$$

上記の2項目の式で求めた値を加算することで、参照データが重なることなく格納できるバッファサイズが得られる。この式にx_Size=5、y_Size=5、z_Size=16を代入し784という値を算出する。このバッファサイズで図6~7で示した格納操作をする。

【0023】図8は、図3に示す3次元数値データの

x, y, zの各方向サイズを256とし、3次元数値データ3-1-1から始まり、z方向を16、x方向を5、y方向を5とした直方体内のデータをバッファに書き込み格納した場合のバッファ内におけるデータの配置状況を示したものである。この場合、図3において、3-1-1~3-1-4の3次元数値データは1番上の(y=0)のxz面のデータであり、3-1-5の3次元数値データは2番目の(y=1)のxz面のデータであり、同様に3-1-5の1段下の3次元数値データは3番目の(y=2)のxz面のデータであり、以下同様である。データが1番上の(y=0)のxz面に属することをy方向が「256²*0」であると表わし、データが2番目の(y=1)のxz面に属することをy方向が「256²*1」であると表わし、以下同様に、256²*2、256²*3、256²*4で表わす。また、3-1-1から始まるy=0での16個のデータ、3-1-5から始まるy=1での16個のデータ、...y=4での16個のデータをxz方向が0~15(番地)のデータとして表わし、3-1-3から始まるy=0での16個のデータ、...y=4での16個のデータをxz方向が256~271(番地)のデータとして表わし、3-1-3の右隣の位置から始まるy=0での16個のデータ、...y=4での16個のデータをxz方向が512~527(番地)のデータとして表わし、さらにその右隣の位置から始まるy=0での16個のデータ、...y=4での16個のデータをxz方向が768~783(番地)のデータとして表わし、さらにその右隣の位置から始まるy=0での16個のデータ、...y=4での16個のデータをxz方向が1024~1028(番地)のデータとして表わす。そして、図8において、xz方向が0~15で、y方向が256²*0である欄は3次元数値データの0~15(3-1-1から始まる16個のデータ)に対応しており、この欄には、3次元数値データの0~15がバッファに格納された場合のそのバッファ内での格納位置情報が書き込まれる。このバッファ内での格納位置(アドレス)は、データの共有メモリアドレスの値をバッファサイズの値で割った余りの値である。上記の場合、データの共有メモリアドレスの値は0~15であるから、これをバッファサイズの値784で割った余りの値は0~15であり、この値0~15が、図8の表のxz方向が0~15で、y方向が256²*0である欄に記入されている。また、3-1-3で始まる16個の3次元数値データ場合は、そのデータの共有メモリアドレスの値は256~271であるから、この値をバッファサイズの値784で割った余りの値は256~271であり、この値256~271がバッファ内での格納位置(アドレス)であり、この値が図8の表のxz方向が256~271で、y方向が256²*0である欄に記入されている。さらに、3-1-5で始まる16個の3次元数値データ

場合は、そのデータの共有メモリアドレスの値は65536~65551であるから、この値をバッファサイズの値784で割った余りの値は464~479であり、この値464~479がバッファ内での格納位置(アドレス)であり、この値が図8の表のxz方向が0~15で、y方向が256²*1である欄に記入されている。

【0024】この結果からすべて異なるバッファ位置に格納されていることが判る。なお、(Size_x, Size_y, Size_z)=(5, 5, 32)、

(5, 5, 48)、(7, 7, 16)なども同様にすべて異なるバッファ位置に格納されることは確認済みである。また、z>0の場合でも隣あうデータとの距離は同じなので同様の結果が得られる。上記までの手段で求めたバッファ構成とサイズを持つバッファをローカルメモリに確保することにより、例えば、角度(0°, 0°, 0°)において25回の共有メモリのリードアクセスによって、784バイトのサイズのバッファの中に13ボクセル分の計算で参照するデータ174個をローカルメモリに持つことができる。これは624回の参照分のデータであり、共有メモリよりローカルメモリが5倍速いとする約4倍の高速化が図れる。

【0025】図9は、ルートプロセッサが持つ全プロセッサの稼働状況管理テーブル1-4-3である。管理テーブル9-1は、接続されているプロセッサの数までの値をとるインデックス部9-2と、稼働可能の通知を通信してきたプロセッサ上の描画プロセスの一意な識別子を格納する部分9-3から成る。ルートプロセッサは接続されているプロセッサの数を得てこの管理テーブルを作成する。識別子部分9-3の初期値は-1が入っている。各プロセッサはローカルメモリが確保でき稼働可能になったならばルートプロセッサに稼働可能の通知を送り、ローカルメモリが確保できない場合は稼働不可能の通知を送る。ルートプロセッサは送られて来た内容を判定し、稼働可能の通知を送ってきたプロセスの識別子を管理テーブルのプロセス識別子9-3に格納する。接続されている全プロセッサの通知を受けた後、稼働不可能の通知を送ってきたプロセッサが行うはずだった処理を稼働可能なプロセッサ数で再分割し、稼働可能なプロセスの識別子を用いて配送する。

【0026】図10は、各プロセッサでの処理において、対象3次元数値データを格納するバッファをローカルメモリに確保する処理の流れを示したものである。ユーザからの要求があると描画処理であるかを判定し(10-1)、描画処理でないならば対応する処理を実行する(10-2)。描画処理ならばルートプロセッサは処理の分割を行ない(10-3)、分担処理の情報・描画条件情報・3次元数値データの全体サイズ・構成情報等を付加した描画要求を各プロセッサに渡し(10-4)、各プロセッサは要求を受け取る(10-5)。次に管理情報テーブルのバッファサイズ2-17が0かを

判定し(10-6)、0であれば1回目のボクセル計算でバッファのサイズを算出する(10-7)。ローカルメモリの空き領域サイズを入手し(10-8)、バッファサイズを算出してバッファを確保する(10-9)。その後確保したバッファを用いて描画処理を行なう(10-10)。

【0027】図11は、バッファの使用制御の処理の流れである。各プロセッサは割り当てられた画素を求めるのに参照するデータの共有メモリアドレスを得て(11-1)、その値をバッファサイズで割った余りでバッファのどの位置に格納するかを求める(11-12)。バッファ管理情報テーブル(図6)を参照し該当位置すでにそのデータがあるかを判定し(11-3)、あればバッファからその内容を参照し(11-4)、なければ共有メモリから読み込んで参照する(11-5)。読み込んだデータは、事前に算出したバッファの位置に格納し(11-6)、アドレスを管理テーブルに格納する(11-7)。同じインデックス値を持つデータが来た場合には、重ね書きをする。

【0028】図12は、描画計算処理中に、計算に必要なローカルメモリが確保できない場合のプロセッサにおける処理の流れである。バッファサイズ削減の有無を識別する削減フラグを用意し0で初期化する(12-1)。ローカルメモリの空き領域を求め(12-2)、現在の指定されたバッファサイズでバッファが確保できるかを判断し(12-3)、可能であれば確保して(12-4)、描画処理を実行する(12-5)。現在のバッファサイズで確保できない場合はローカルメモリの空き領域サイズと規定した最少バッファサイズとを比較し(12-6)、最少バッファサイズよりも大きい場合は最少バッファサイズと空き領域サイズとの間の値でバッファサイズを求めバッファを確保する(12-7)。この時求めたバッファのx、y、z方向サイズは管理情報テーブルに再登録し(12-8)、削減フラグに1を代入する(12-9)。得たバッファを使って描画処理を実行する(12-5)。ローカルメモリの空き領域サイズが最少バッファよりも小さい場合には処理の分割を行なっているルートプロセッサにその旨を通知し(12-10)、処理を終了する。ルートプロセッサでは通知を受けると(12-11)、該当するプロセッサの担当処理を稼働可能なプロセッサ数で分割し(12-12)、稼働可能なプロセッサに再分配する(12-13)。ルートプロセッサも処理を分担し描画処理を行なう(12-14)。他のプロセッサは通知を受け取り(12-15)、描画処理を行なう(12-16)。この処理により、いずれかのプロセッサでローカルメモリを確保できなくても処理を続けることができる。

【0029】図13はバッファのz方向サイズの再算出処理の流れを示したものである。図10の(10-10)にある描画処理中に、1ピクセルの処理で参照する

データのz方向の最大サイズ(レイ・キャストリングにおける視線トレース回数)を算出する(13-1)。算出したz方向最大サイズがバッファのzサイズよりもデータ読み込み単位数以上に大きいかを判断し(13-2)、大きい場合には次に削減フラグが0でメモリに余裕があるかを判断し(13-3)、0であれば描画処理終了後にバッファのz方向のサイズを算出した値よりも大きくかつ一番近いデータ読み込み単位数の倍数に置き換える(13-4)。削減フラグが1の場合には何もしない。また、算出したz方向最大サイズがバッファのzサイズよりもデータ読み込み単位数以上に小さい場合にも、描画処理終了後にバッファのz方向のサイズを算出した値よりも大きくかつ一番近いデータ読み込み単位数の倍数に置き換える(13-4)。この値は次の計算時に最適なバッファサイズを算出するために使用する。

【0030】図14は本発明の第2実施例であり、本発明を適用した別システムの例である。本実施例は第1実施例で示した構成をサブシステムとし、ホストプロセッサ14-1と接続する構成とする。ホストプロセッサ14-1には描画処理を分割し各プロセッサに分配・通信する処理14-1-1がある。また、ホスト側のメモリ14-2には接続されているプロセッサの稼働状況情報14-2-1がある。ホスト側は分配・通信処理14-1-1で処理をプロセッサ1-1~3に分配し、各プロセッサ1-1~3はその内容を受信して描画処理を行なう。描画処理とバッファに関する処理は第1の実施例と同じである。但し第1の実施例と異なり、ローカルメモリにバッファが確保できない場合には、ホストプロセッサ14-1にその旨の通知を送る。各プロセッサ1-1~3はその内部にこの手段1-1-3、1-2-3、1-3-3を持つ。ホストプロセッサ14-1はこの通知を受信し稼働可能な接続プロセッサの数で処理を再分割し稼働可能なプロセッサに分配送信する手段14-1-2を持つ。この手段4-1-2を用いて処理を再分配する。各プロセッサ1-1~3は再分配された処理を受信する。受信後の処理は第1実施例と同じである。

【0031】

【発明の効果】本発明は参照するデータのみローカルメモリに読み込む方式のため、よけいなデータ転送オーバーヘッドがない。またすべての参照データを格納するバッファが確保できない場合でも、参照データの配置情報に適したサイズでバッファを確保するため、より小さなサイズで実現でき、かつデータの再参照時のヒット率も高くなり、再参照時のアクセス時間が短縮でき、高速な処理を実現できる。

【図面の簡単な説明】

【図1】本発明の実施例のシステムの構成を示す図である。

【図2】管理情報テーブルの例を示す図である。

【図3】3次元数値データの構成例を示す図である。

13

【図4】1ボクセルのボリュームレンダリング計算処理での3次元角度(0°, 0°, 0°)での参照パターン例を示す図である。

【図5】求める画素と参照するデータ群との単純な関係を示す図である。

【図6】バッファ管理情報テーブルの例を示す図である。

【図7】確保したバッファと共有メモリ内の3次元数値データとの対応を示す図である。

【図8】算出したバッファサイズに基づく3次元数値データのバッファへの格納結果を示す図である。

【図9】稼働状況管理テーブルの例を示す図である。

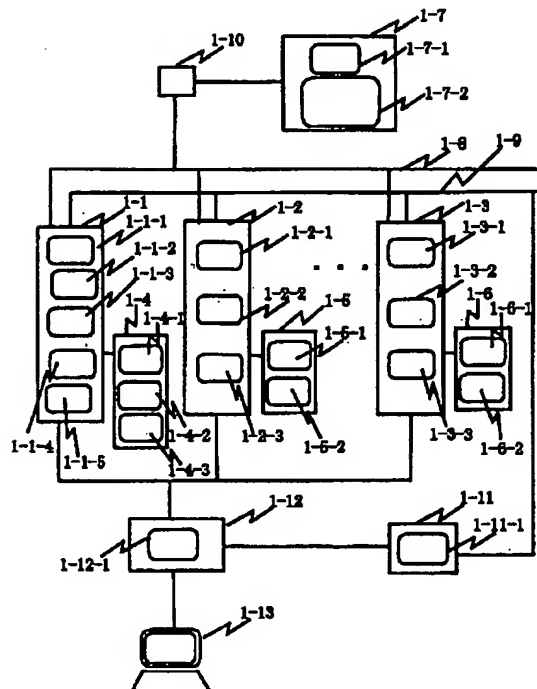
【図10】バッファ確保処理のフローチャートを示す図である。

【図11】バッファ使用制御のフローチャートを示す図である。

【図12】ローカルメモリが確保できない場合の処理の

【図1】

図 1



14

フローチャートを示す図である。

【図13】バッファのz方向サイズ再算出のフローチャートを示す図である。

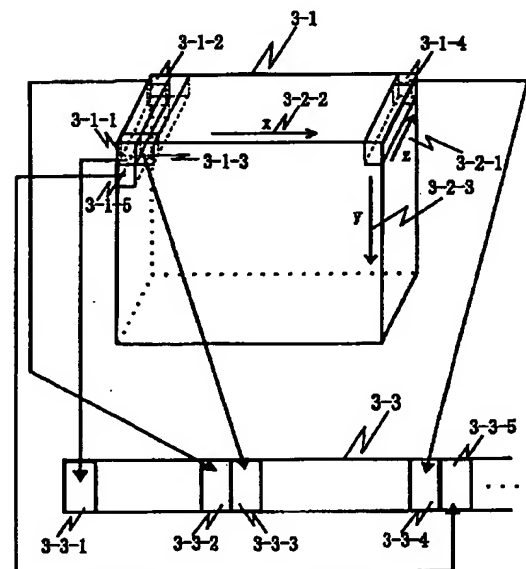
【図14】第2の実施例のシステム構成を示す図である。

【符号の説明】

- 1-1 ルートプロセッサ
- 1-2~3 処理プロセッサ
- 1-4~6 ローカルメモリ
- 1-7 共有メモリ
- 1-8~9 バス
- 1-10 バスブリッジ
- 1-11 フレームバッファ
- 1-12 表示プロセッサ
- 1-13 表示装置
- 14-1 ホストプロセッサ
- 14-2 ホスト側のメモリ

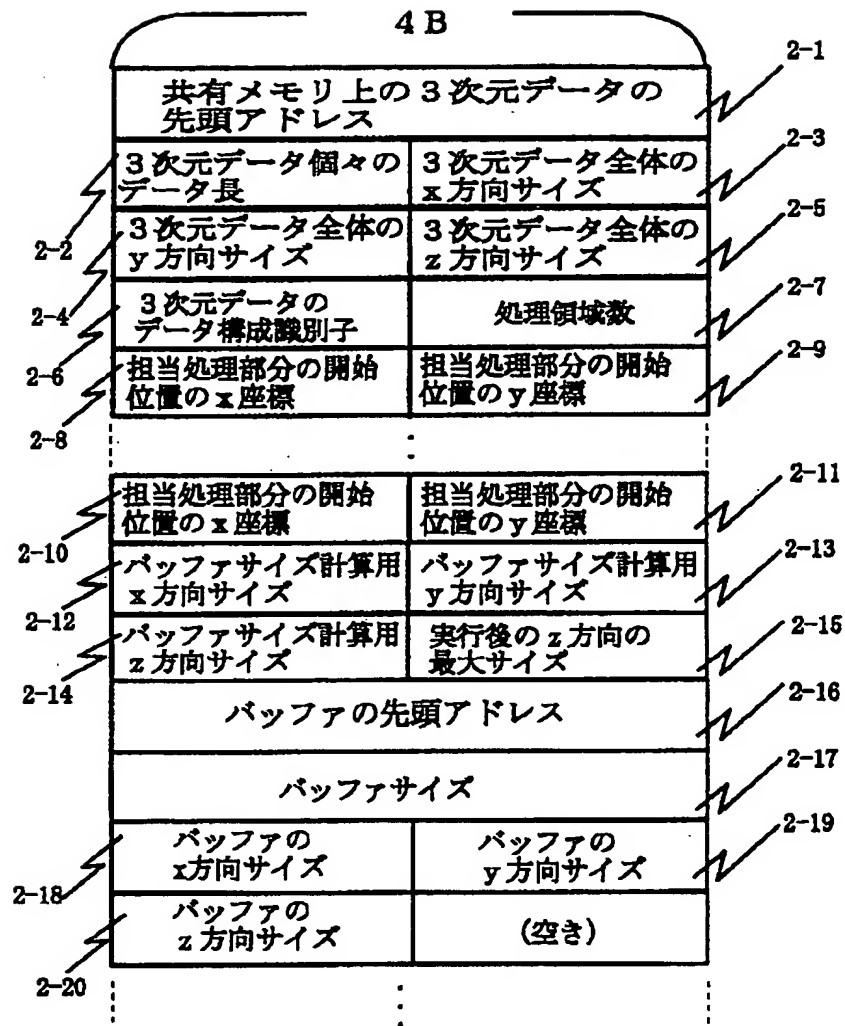
【図3】

図 3



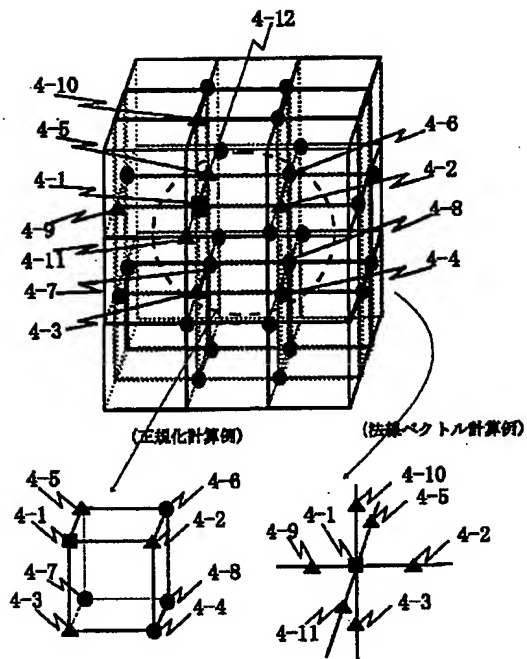
【図2】

図 2



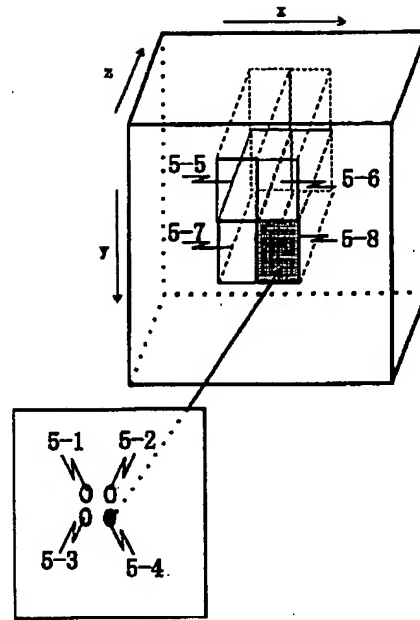
【図4】

図 4



【図5】

図 5



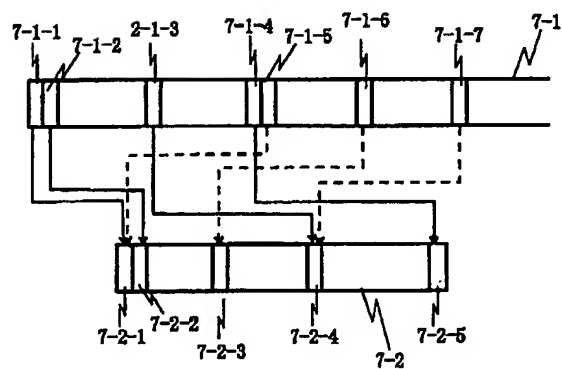
【図6】

図 6

	6-1	6-2
index1	addr1	
index2	addr2	
index3	-1	
index4	-1	
	⋮	

【図7】

図 7



【図8】

図 8

8-2

8-1

7方向 6x方向	256 ² *0	256 ² *1	256 ² *2	256 ² *3	256 ² *4
0~15	0~15	464~479	114~129	608~623	288~303
256~271	256~271	720~735	370~385	80~95	544~559
512~527	512~527	192~207	626~641	338~353	16~31
768~783	768~783	448~463	98~113	592~607	272~287
1024~1028	240~255	704~719	354~369	84~99	528~543

8-3

【図9】

図 9

9-2

9-1

9-3

Index	稼働可能なプロセッサ上の描画処理プロセス識別子
1	プロセスID
2	プロセスID
3	-1
4	-1
⋮	⋮
接続プロセッサ個数	-1

【図14】

図 14

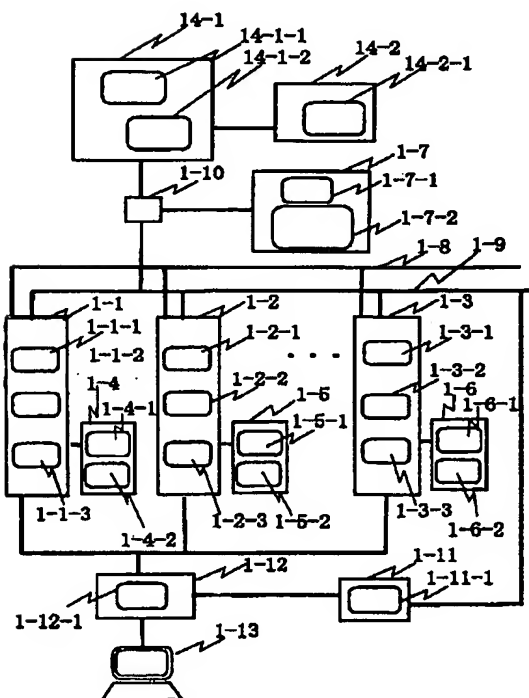
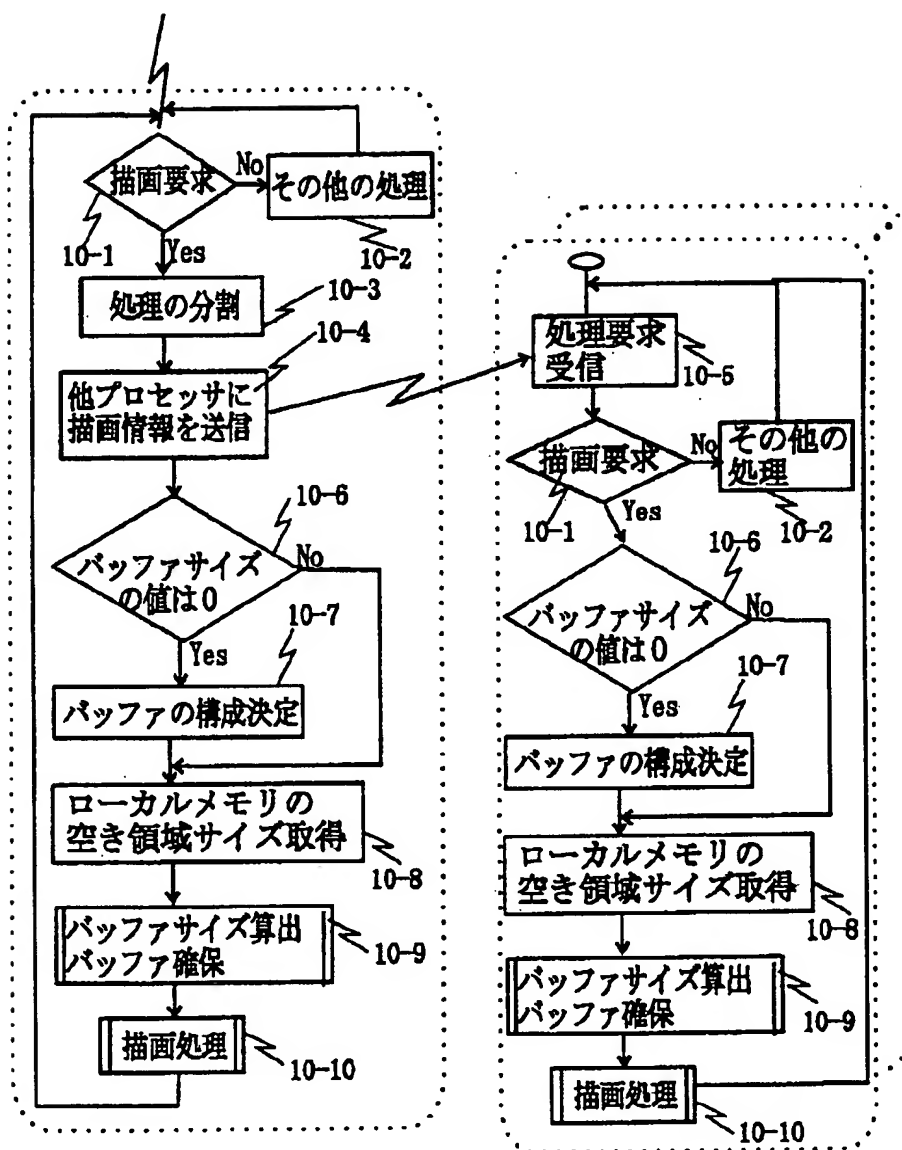
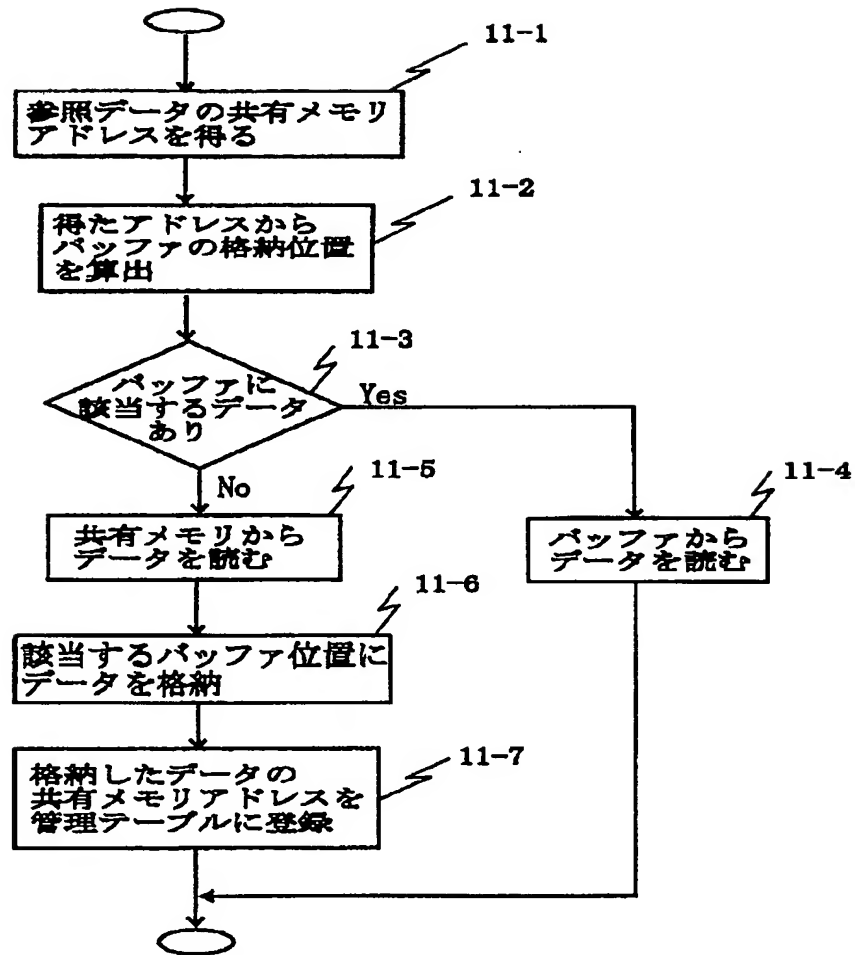


图 10



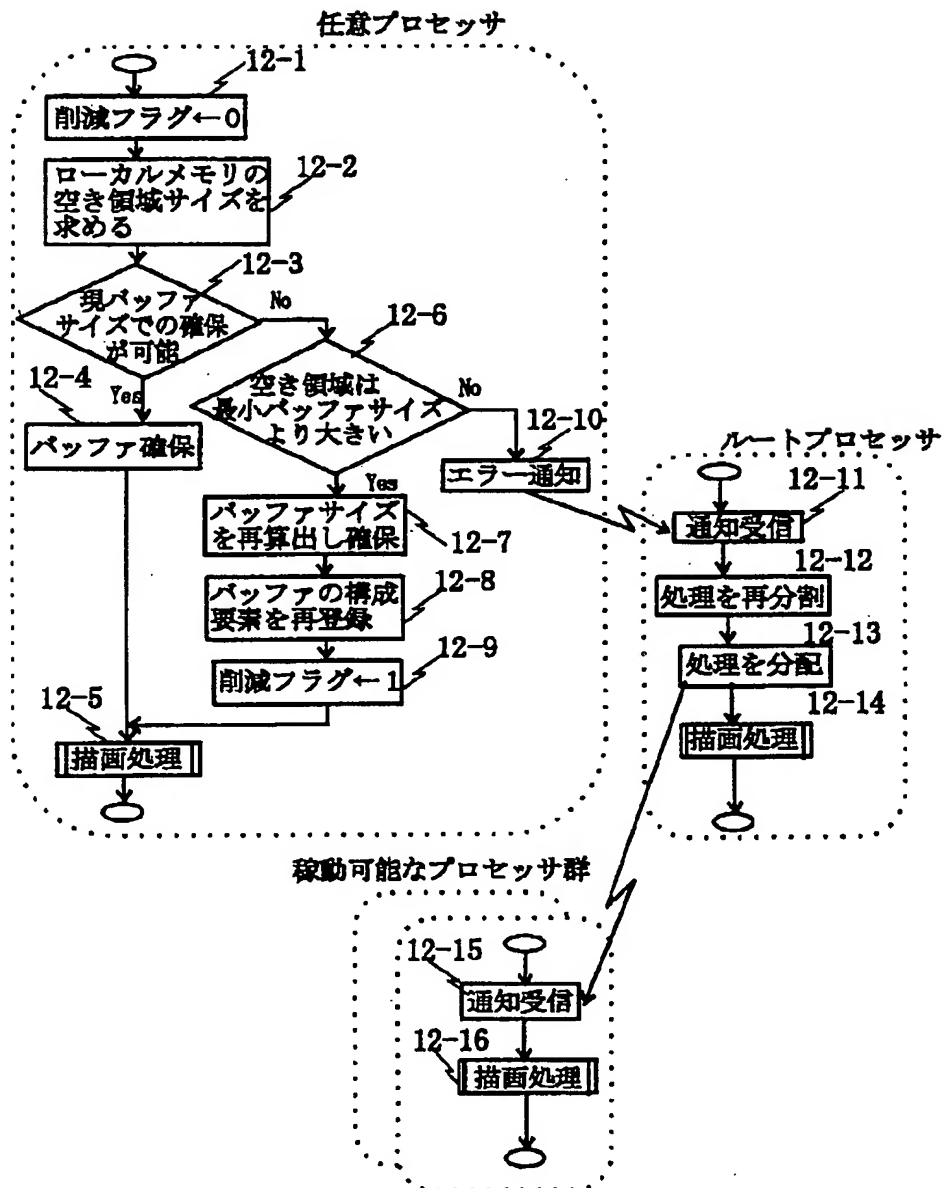
【図11】

図 11



【図12】

図 12



【図13】

図 13

